Introducción a Ruby

Paradigma de programación

Overview

- Paradigma de programación
- Clases
- Cómo crear objetos
- Cómo acceder a los datos dentro de un objeto

Paradigma de programación

- Orientado a objetos
- Bloques inspirados en lenguajes funcionales
- Dinámico
- Débilmente tipado

Revisión de Orientación a Objetos

- Las Clases son como Tipos de Objetos:
  - Contienen métodos que definen su comportamiento.
  - Contienen atributos que definen su estructura.
- Los objetos son instancias de una clase.
  - Los objetos tienen variables de instancia.

Variables de Instancia

- Inician con **@**
  - Ejemplo: **@nombre**
- No son declaradas
  - Se asume su existencia la primera vez que son utilizadas.
- Disponibles para todos los métodos de instancia de la clase.

Creación de objetos

- Las clases son **fábricas**!
  - Llamando al método new se **crea una instancia** de una clase.

> **new** causa la ejecución del método **initialize**

- El estado de un objeto puede (y debe) ser inicializado dentro del método constructor.

Creación de un objeto

```ruby
class Persona
  def initialize (nombre, edad) #CONSTRUCTOR
    @nombre = nombre
    @edad = edad
  end
  def get_info
    @informacion_adicional = "Interesante"
    "Nombre: #{@nombre}, edad: #{@edad}"
  end
end

Persona1 = Persona.new("Juan", 14)

p Persona1.instance_variables # [:@nombre, :@edad]
puts Persona1.get_info # => Nombre: Juan, edad: 14

p Persona1.instance_variables
# [:@nombre, :@edad, :@informacion_adicional]
```

Acceso a datos

- **Las variables** de instancia son **privadas**.
  - No pueden ser accedidas fuera de la clase.
- **Los métodos** tienen el modificador “**public**” por defecto.
- Para acceder a las variables de instancia se deben definir métodos “**getters**” y “**setters**”

Definición de acceso manual

```ruby
class Persona
  def initialize (nombre, edad) #CONSTRUCTOR
    @nombre = nombre
    @edad = edad
  end
  def get_info
    @informacion_adicional = "Interesante"
    "Nombre: #{@nombre}, edad: #{@edad}"
  end
  def nombre
    @nombre
  end
  def nombre=(nuevo)
    @nombre = nuevo
  end
end
```

```ruby
persona1 = Persona.new("Juan", 14)
puts persona1.nombre # Juan
persona1.nombre = "Mike"
puts persona1.nombre # Mike
```

Acceso a datos

- Es común que la lógica de getter/setter sea **simple**.
  - Obtener el valor existente / Establecer un valor nuevo.
- Entonces, Ruby propone una forma **más simple** en lugar de definir cada uno de los métodos setters y getters.

La forma fácil

- Use **attr_*** form instead
  - **attr_accessor – getter y setter**
  - **attr_reader – getter solamente**
  - **attr_writer – setter solamente**

Attr accessor

```ruby
class Persona
  attr_accessor :nombre, :edad
end

persona1 = Persona.new
p persona1.nombre # => nil
persona1.nombre = "Marcos"
persona1.edad = 15
persona1.edad = "fifteen"
puts persona1.edad # => fifteen
```

Acceso a datos

Existen dos problemas con el ejemplo anterior:

- Persona está en un **estado no inicializado** después de la creación.
  - No tiene nombre ni edad.
- Probablemente queramos **controlar** la “edad” máxima asignada.

Acceso a datos

- Solución: utilizar un **constructor** y un método de **seteo** para “edad” **más inteligente**.

Primero vamos a hablar de **self**..

Self

- **Dentro de un método de instancia**, **self** se refiere al objeto en sí.
  - Usualmente, utilizar **self** para llamar a otros métodos de la misma instancia es extraño.
- En otras ocasiones, usar **self** es obligatorio.
  - Ejemplo: cuando la sentencia implica la asignación de una variable local.
- **Fuera de un método de instancia**, **self** se refiere a la clase en sí.

Self

```ruby
class Persona
  attr_reader :edad
  attr_accessor :nombre

  def initialize (nombre, edadVar) # CONSTRUCTOR
    @nombre = nombre
    self.edad = edadVar # call the edad= method
    puts edad
  end
  def edad= (nueva_edad)
    @edad = nueva_edad unless nueva_edad > 120
  end
end

persona1 = Persona.new("Kim", 13) # => 13
puts "Mi edad es #{persona1.edad}" # => Mi edad es 13
persona1.edad = 130 # Se intenta cambiar la edad
puts persona1.edad # => 13 (El seter no aplicó el cambio)
```

Entonces..

- Los objetos son **creados** con **new**
- La forma corta de **getter**/**setter** es **attr_***
- No olvidar cuando **self** es necesario.